クレジット:
UTokyo Online Education 統計データ解析 Ⅱ 2018 小池祐太

東京大学 THE UNIVERSITY OF TOKYO

CC BY NC ND

統計データ解析 (II) 第 4 回

小池祐太

2018年 5 月 10 日

1. データの抽出
2. ファイルを用いたデータの読み書き
3. 記述統計量によるデータの要約
4. データの可視化
   - 基本的な描画
   - ヒストグラム
   - 箱ひげ図
   - 相関行列の可視化
   - その他の描画関数

データの抽出

- データフレームから必要な部分集合を取り出す際に複雑な条件を指定する場合, 添え字を指定するのではコードが読みにくくなってしまう
- そのような場合にも対応できるように, 関数 `subset()` が用意されている
- 関数 `subset()` の基本書式

```
subset(x, subset, select, drop = FALSE)
```

  - `x`: データフレーム
  - `subset`: 抽出したい行に関する条件
  - `select`: 抽出したい列に関する条件 (未指定の場合はすべての列が抽出される)
  - `drop`: 結果が 1 行もしくは 1 列のデータフレームになる場合に, 結果をベクトルとして返すか否か

- 実行例 `subset.r`

ファイルを用いたデータの読み書き

- 実際の解析の過程においては，収集されたデータを読み込んだり，整理したデータを保存したりする必要が生じる
- R では一般に用いられる CSV 形式 (comma separated values) のテキストファイルと，R の内部表現を用いたバイナリーファイル (ここでは RData 形式と呼ぶ) をサポートしている
- 以下では，データフレームを対象として，それぞれの形式でファイルの読み書きを行うための関数を纏める

作業ディレクトリの確認と変更

- R の実行は特定のフォルダ (ディレクトリ) 上で行われており, そのフォルダを**作業ディレクトリ**と呼ぶ
- R のコード内でファイル名を指定した場合, 特に指定しない限り作業ディレクトリに存在するものとして扱われる
- 現在の作業ディレクトリは, RStudio のコンソールの上部, もしくは

  ```
  getwd()
  ```

  で確認できる

作業ディレクトリの確認と変更

- 作業ディレクトリの変更には関数 `setwd()` を利用するか, RStudio 上部の「Session」という項目から「Set Working Directory」を選び, その中の「Choose Directory...」という項目を選択すれば, 変更後のフォルダを選択できるようになる
- 実行例 `getwd.r`

ファイルを用いたデータの読み書き

- 1 つのデータフレームを CSV 形式のファイルへ書き出すには，関数 `write.csv()` を用いる
- 基本書式

```
write.csv(x, file = "")
```

  - `x`: 書き出したいデータフレーム
  - `file`: 書き出すファイルの名前
  - 他にも細かいオプションあり．ヘルプ参照
- ファイルの保存先は (指定しない限り) 作業ディレクトリとなる
- 実行例 `data-write.csv.r`

ファイルを用いたデータの読み書き

- CSV 形式のファイルから読み込むには，関数 `read.csv()` を用いる
- 基本書式

```
read.csv(file, header = TRUE, row.names)
```

  - `file`: 読み込みたいファイルの名前 (作業ディレクトリ下にある必要あり．もしくはディレクトリも指定)
  - `header`: ファイルの 1 行目をデータフレームの列名として使うか否か?
  - `row.names`: データフレームの行名を指定．(i) 行名を含む列番号/列名を指定, (ii) 行名の直接指定, というオプションがある．デフォルトでは行番号がそのまま行名になる．
  - 他にも細かいオプションあり．ヘルプ参照
- なお, より一般のテキストファイルを読み込むための関数として `read.table()`, `scan()` などがある

ファイルを用いたデータの読み書き

- 以降の講義の実行例で利用するデータ `kikou2016.csv` は, 以下の Web ページ

  `https://elf-c.he.u-tokyo.ac.jp/courses/228`

  からダウンロードできる
- 実行例 `data-read.csv2.r`

ファイルを用いたデータの読み書き

- RData 形式のファイルへの書き出しは，関数 `save()` を用いる
- CSV 形式と異なり，複数のデータフレームを 1 つのファイルに同時に保存することもできる
- 基本書式

```
save(..., file)
```

  - `...`: 保存したいオブジェクト名 (複数可, データフレーム以外も可)
  - `file`: 書き出すファイルの名前

- ファイルの保存先は (指定しない限り) 作業ディレクトリとなる
- 実行例 `data-save.r`

ファイルを用いたデータの読み書き

- RData 形式のファイルからの読み込みは，関数 `load()` を用いる
- 基本書式

```
load(file)
```

  - `file`: 読み込みたいファイルの名前 (作業ディレクトリ下にある必要あり．もしくはディレクトリも指定)

- 実行例 `data-load.r`

記述統計量によるデータの要約

- データ解析の出発点は, 与えられたデータ全体の特徴や傾向を把握することである
- そのための基本的な方法の 1 つは, データの特徴を適切に表す統計値を計算することである
- そのような統計値を記述統計量, 要約統計量もしくは基本統計量と呼ぶ

記述統計量によるデータの要約

- N 個のデータ $x_1, x_2, \ldots, x_N$ が与えられたとき, それらを代表する値として, **平均 (mean)**

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{x_1 + x_2 + \cdots + x_N}{N}$$

  が頻繁に利用される
- 平均は R では関数 `mean()` で計算できる

記述統計量によるデータの要約

- また, データのばらつき具合の指標として, **分散 (variance)**

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \cdots + (x_N - \bar{x})^2}{N-1}$$

およびその平方根である**標準偏差 (standard deviation)**

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

が広く利用されており, それぞれ関数 `var()` および関数 `sd()` で計算できる

記述統計量によるデータの要約

- データの順位にもとづく記述統計量もよく利用される
- 例えば, $x_1, \ldots, x_N$ の**最大値 (maximum)** は関数 `max()` で, **最小値 (minimum)** は関数 `min()` でそれぞれ計算できる

記述統計量によるデータの要約

- データを

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)}$$

のように昇順に並べ替えた際に中央の位置にくる値を**中央値**または**メディアン (median)** と呼ぶ
- N が奇数の場合, 中央値は $x_{((N+1)/2)}$ であり, N が偶数の場合は $(x_{(N/2)} + x_{(N/2+1)})/2$ である
- 中央値は関数 `median()` で計算できる
- 中央値は平均と同様データを代表する値だと考えられるが, 平均と比較して, 計算結果がデータに含まれる異常な値 (**外れ値 (outlier)** と呼ばれる) の影響を受けにくい

記述統計量によるデータの要約

- 中央値の一般化として, $\alpha \in [0, 1]$ に対して, その点以下のデータの個数が全体の**約** $100\alpha\%$になるような点を $100\alpha\%$**分位点 (quantile)** と呼ぶ
- 特に 25%分位点および 75%分位点をそれぞれ**第 1 四分位点**, **第 3 四分位点**と呼ぶ
  - **第 2 四分位点**は 50%分位点となるが, これは中央値のことである
- ベクトル `x` の 100`alpha`%分位点は `quantile(x, alpha)` で計算できる
- 分位点は一意的には定まらず, いくつかの計算方式がある: `help(quantile)` 参照
- 実行例 `descriptive.r`

記述統計量によるデータの要約

- データフレームが与えられた際には, 列 (あるいは行) ごとに記述統計量を計算したい状況が頻繁にある
- そのような計算に便利な関数として関数 `apply()` がある. 関数 `apply()` は基本的に以下のような書式で利用する:

```
apply(X, MARGIN, FUN)
```

  - `X`: データフレーム
  - `MARGIN`: 行ごとの計算には 1 を, 列ごとの計算には 2 を指定
  - `FUN`: 求めたい統計量を計算するための関数
- なお, データフレーム `x` に対して `summary(x)` を実行することで, 列ごとの最小値, 第 1 四分位点, 中央値, 平均, 第 3 四分位点, 最大値がそれぞれ計算される
- 実行例 `apply.r`

記述統計量によるデータの要約

- 複数のデータが与えられた場合, それらのデータの間の関係性を知りたい場合が頻繁に生じる
- そのような目的のための最も基本的な記述統計量に**相関係数 (correlation coefficient)** がある
  - 2 種類のデータ間の比例関係の大きさを計測
- 2 種類のデータ $x_1, x_2, \ldots, x_N$ および $y_1, y_2, \ldots, y_N$ に対して, それらの相関係数は

$$\rho = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{1}$$

で定義される ($\bar{x}$ および $\bar{y}$ はそれぞれ $x_1, x_2, \ldots, x_N$ および $y_1, y_2, \ldots, y_N$ の平均)

記述統計量によるデータの要約

- 相関係数は -1 以上 1 以下の値をとり, 1 に近いほど正の比例関係が強く, -1 に近いほど負の比例関係が強いことになる
- なお, (1) の分子の統計量を $N-1$ で割ったものは**共分散 (covariance)** と呼ばれる
- 相関係数および共分散はそれぞれ関数 `cor()` および関数 `cov()` で計算できる
  - 2 種類のデータ `x` および `y` が与えられたとき, それらの相関係数は `cor(x,y)` で計算できる
  - `x` がデータフレームのとき, `cor(x)` は (i,j) 成分が `x` の i 列と j 列の間の相関係数であるような行列 (**相関行列 (correlation matrix)**) を計算
  - 共分散についても同様
- 実行例 `cor.r`

データのプロット

- データ全体の特徴や傾向を把握するために効果的な方法は, データの可視化である
- R にはきわめて多彩な作図機能が用意されており, ここではいくつかの代表的な描画関数を取り上げて解説する
- 描画関連の関数は色, 線種や線の太さ, あるいは図中の文字の大きさなどを指定するために, 多彩なオプションを用意しており, ここでは説明しきれないため, 必要に応じて関数 `help()`(ヘルプの表示) と `example()`(例題の表示) を参照

基本的な描画

- 描画において基本となるのは関数 `plot()` である
- 基本書式 (ベクトルの描画)

```
plot(x, type = "p", xlim = NULL, ylim = NULL, main =
    NULL, xlab = NULL, ylab = NULL, ...)
```

  - `x`: ベクトル
  - `type`: 描画タイプ. デフォルトは`"p"` (点プロット). 他に`"l"` (折れ線プロット) などがある
  - `xlim`: x 軸の範囲. デフォルトでは自動的に決定
  - `ylim`: y 軸の範囲. デフォルトでは自動的に決定
  - `main`: 図のタイトル. デフォルトではなし
  - `xlab`: x 軸のラベル名. デフォルトでは `Index` となる
  - `ylab`: y 軸のラベル名. デフォルトでは `x` のオブジェクト名
  - `...`: 他のオプション. 詳細は `help(par)` 参照

基本的な描画

- よく利用される plot のオプション
  - col: 描画するデータの色の指定．"red"や"blue"など．指定することのできる色の名前は関数 colors() で照会できる
  - pch: 描画される点の形．数字で指定．詳細は help(points) 参照
  - cex: 描画される文字の大きさ．デフォルトの何倍にするかで指定
  - lty: 描画される線のタイプ．実線, 破線など．タイプ名もしくは数字で指定．詳細は help(par) 参照
  - lwd: 描画される線の太さ．数字で指定
- ベクトル x に対して plot(x) を実行すれば, 横軸に成分番号, 縦軸に各成分を描画した点プロットが作成される
- 実行例 plot3.r

基本的な描画 (関数)

- 1 変数関数の描画も関数 `plot()` で可能
- 基本書式 (1 変数関数の描画)

  ```
  plot(x, y = 0, to = 1, ...)
  ```

  - `x`: 1 変数関数
  - `y`: x 軸の左端
  - `to`: x 軸の右端
  - `...`: “ベクトルの描画” と同じオプションが利用可能
- 別の関数 `f` を重ね書きをしたい場合,

  ```
  curve(f, add = TRUE, ...)
  ```

  を実行してやればよい (... には “ベクトルの描画” と同じオプションが利用可能)
- 実行例 `plot3.r`

基本的な描画 (散布図)

- 2 種類のデータ $x_1, \ldots, x_N$ および $y_1, \ldots, y_N$ が与えられたとき，点 $(x_1, y_1), \ldots, (x_N, y_N)$ を平面上に描画した図を**散布図 (scatter plot)** と呼ぶ
- 散布図も関数 `plot()` で作成できる
- 基本書式 (散布図)

```
plot(x, y = NULL, ...)
```

  - x: 1 種類目のデータ $x_1, \ldots, x_N$
  - y: 2 種類目のデータ $y_1, \ldots, y_N$
  - ...: “ベクトルの描画” と同じオプションが利用可能

基本的な描画 (散布図)

- また, データフレーム x の変数 A と変数 B に関して散布図を作成したい場合, コマンド

  ```
  plot(B ~ A, data = x, ...)
  ```

  も利用できる
- 実行例 `plot3.r`

基本的な描画 (凡例)

- 関数 `legend()` によってグラフに凡例を追加することができる
- なお, 以下の実行例で見るように, R には数式を扱う機能がある. 詳細は `help(plotmath)` を参照
- 実行例 `legend.r`

基本的な描画 (日本語を含む図の描画)

- OS によっては日本語を含む図を描画すると文字化けする場合がある
- その場合, 関数 `par()` のオプション `family` に適当なフォントファミリーを指定することで文字化けを回避できる場合がある
- 例えば, Mac OS のデフォルトの設定では日本語を含む図は文字化けしてしまうが, 以下のコマンドをコンソール上で実行することで文字化けを回避できる

```
par(family = "HiraginoSans-W4")
```

  - フォントファミリーとしてヒラギノ角ゴシック W4 を指定している (数字を変えると太さが変わる)

- 実行例 `plot-kion.r`

ヒストグラム

- **ヒストグラム (histogram)**
  - データの値の範囲をいくつかの区間に分割し, 各区間に含まれるデータの個数を棒グラフにしたもの
  - 棒グラフの横幅が区間に対応し, 面積が区間に含まれるデータの個数に比例するようにグラフを作成する
  - データの分布の仕方 (どのあたりに値が集中しているか, どの程度値にばらつきがあるかなど) を可視化するのに有効
- ヒストグラムは関数 `hist()` で作成できる

ヒストグラム

- 基本書式

```
hist(x, breaks, freq)
```

  - `x`: ヒストグラムを描画するベクトル
  - `breaks`: 区間の分割の仕方を指定．数字を指定するとデータ範囲をその数字に近い個数に等分割する．デフォルトの個数は Sturges の公式によって決定される．すなわち，データ数を n とすると，$\lceil \log_2 n + 1 \rceil$ である．[1] その他の指定方法もある (ヘルプ参照)
  - `freq`: `TRUE` 指定すると縦軸をデータ数にし，`FALSE` 指定すると縦軸をデータ数/全データ数とする．デフォルトは `TRUE` (`breaks` の指定によって変わる場合あり)
  - 他にも `plot` で指定できるオプションが利用可能

- 実行例 `hist3.r`

[1] $\lceil x \rceil$ は x 以下の最大の整数を表す．

箱ひげ図

- **箱ひげ図 (boxplot)**
  - データの中心, 散らばり具合および外れ値を考察するための図 (ヒストグラムの簡易版)
  - 複数のデータの分布の比較の際に有効
  - データの第 1 四分位点を下端, 第 3 四分位点を上端とする長方形 (箱) と, 第 1 四分位点, 第 3 四分位点からそれぞれ箱の長さの 1.5 倍以内にあるデータのうちの最小の値, 最大の値を下端, 上端とする直線 (ひげ) からなる
  - ひげの外側のデータは点で表示される
  - 中央値は太線で表示される
- 箱ひげ図は関数 `boxplot()` で描画できる

箱ひげ図

- ベクトル x に対する箱ひげ図は boxplot(x,...) で描画できる (... に関数 plot( ) と同様のオプションを指定可能)
- データフレーム x に対して, boxplot(x,...) は列ごとの箱ひげ図を描画
- データフレーム x において, 変数 A が「分類」を表す変数 (性別, 植物の種類など)[2] の場合, 別の変数 B に対して,

```
boxplot(B ~ A, data = x, ...)
```

  は変数 B を変数 A で分類した場合の, 分類ごとの箱ひげ図を描画する
- 実行例 boxplot2.r

[2] **質的変数**と呼ばれる

相関行列の可視化

- 列数が非常に多い大規模データフレームの変数間の相関の様子を見る場合, 相関行列の可視化が便利である
- パッケージ `corrplot` には相関行列を可視化するための関数 `corrplot()` および関数 `corrplot.mixed()` が用意されている
- 実行例 `corrplot.r`

その他の描画関数

- データフレーム x に対して plot(x,...) もしくは pairs(x, ...) を実行すると, すべての列のペアに対する散布図を行列状に並べた図を作成する
  - 変数 A1, ..., Ak のみ考えたい場合, plot(~ A1 + ··· + Ak, data = x, ...) もしくは pairs(~ A1 + ··· + Ak, data = x, ...) を利用
- ベクトル x の各成分の値に基づく円グラフの作成は, pie(x, ...) で実行できる
- 実行例 graphic-misc2.r